PHYS 5002 Project

Mohamed Elbeltagi

1 Introduction

For problems of classification in particle physics (and data science in general), there exists a set of events belonging to different classes, such as signal and background, and it is required to differentiate the background events from the signal events, for example the events may have features or variables (measurements from some detector) such as energy, position (in x, y, and z), and collected charge, and so we expect the background events to differ from the signal events in these features, but we quickly run into the problem that for events living in some multidimensional vector space with features as its basis, (for the above example it would be a 5 dimensional vector space where the events live) it is very hard to visualize the different classes or imagine what the higher dimensional surface separating them looks like, so a solution to this might be considering each feature individually, and conducting hypothesis tests for each, so looking at energy for example, a decision boundary or cut is drawn to separate background from signal, the problem with this is it becomes tedious to find this cut for each feature separately, and the more the features, the bigger the problem, and even after classifying based on each individual feature, you need some way to combine them. So to deal with this, techniques of dimensionality reduction are employed, where the higher dimensional data are projected onto some lower dimensional subspace where the cut application and hypothesis testing can be done, an illustration of this can be seen below in figure 1 [1].

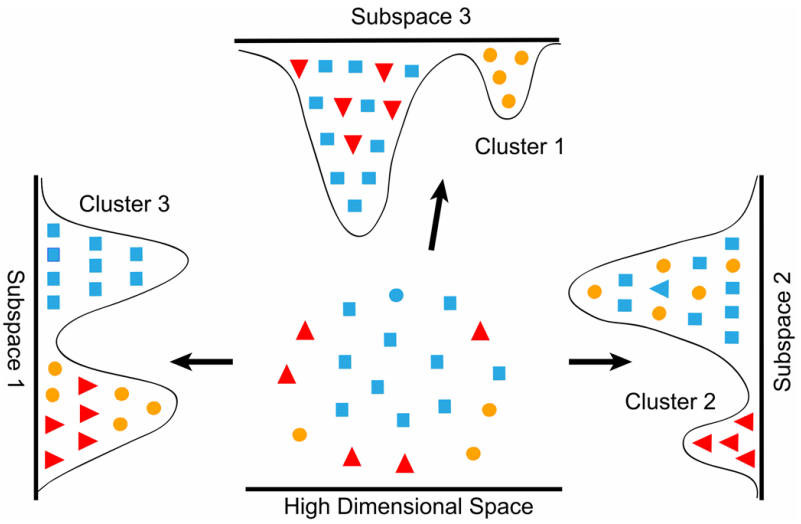

Figure 1: Illustration of projecting higher dimensional data so subspaces where different classes are separable.

For the problem in this project we have training data for the signal and background, and mystery data which we seek to classify into either signal or background, each event is 3 dimensional (a, b, and c), and our method of dimensionality reduction is the Fisher discriminant, which is a linear combination of the features, constructed based on the training data, to get a $1D$ test statistic for hypothesis testing (more on the Fisher discriminant and how it is constructed later in section 3), which can then be applied to the mystery data.

$$t(\vec{x}) = \sum_i k_i x_i = k_1 a + k_2 b + k_3 c \tag{1}$$

2 Training Data Properties

The training data consists of a thousand events each for the background set and the signal set. Each event has three values for a, b, and c, these are the features or variables. Below I show the $1D$ histograms (note: the histograms are not normalized as I'm not interested in the pdf, only the means, to see how close they are, and the standard deviations, to see how sharp each peak is) for each of the variables in the signal and background, to see how similar they are, and which variables could possibly be more useful in discrimination (those that have less overlap, or means that are farther apart), after that I show $2D$ scatter plots of the variables, then a table listing the covariance matrix values, and correlations for signal, and background.

As we can see below (in figures 2, 3, 4) the means for a are closer together (i.e. more overlap) than b and c, with b having the largest separation, on the other hand a has the smallest standard deviation, so the peaks are more sharply defined, as we will see in the next section 3 the weights k_i will depend on the difference of the means of background and signal, and the inverse of the variances.

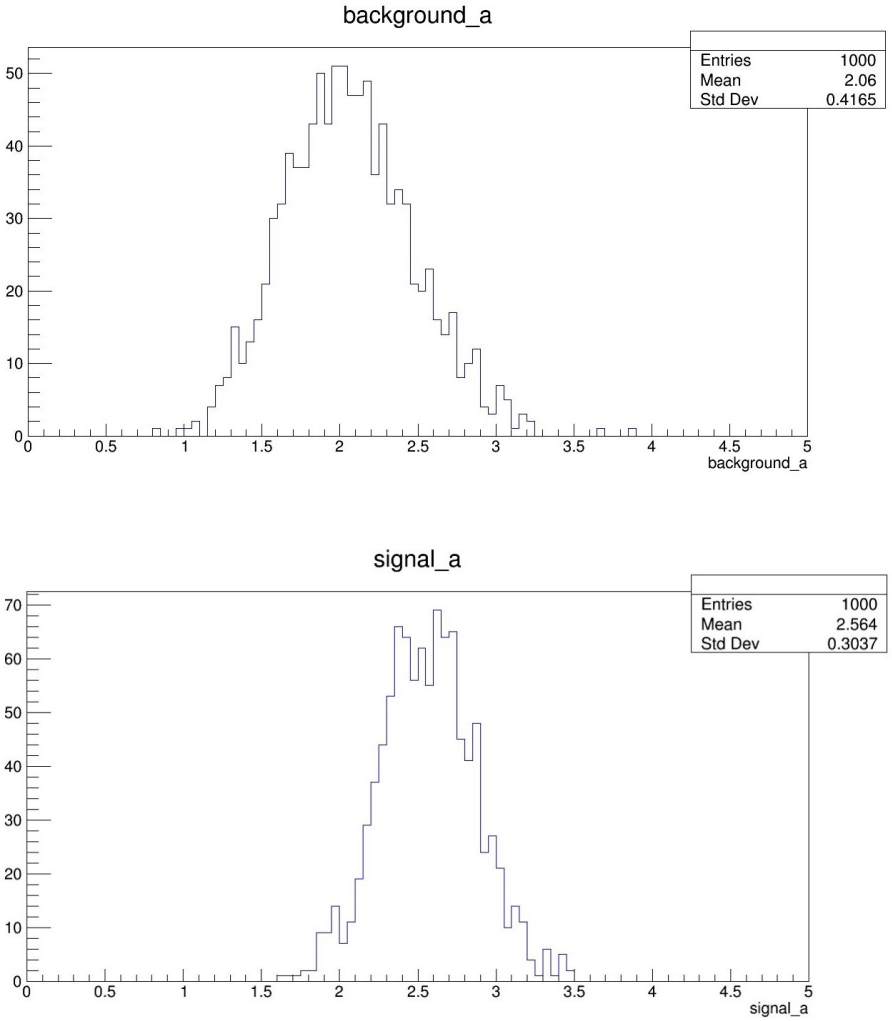

Figure 2: Top: Histogram for background a. Bottom: Histogram for signal a.

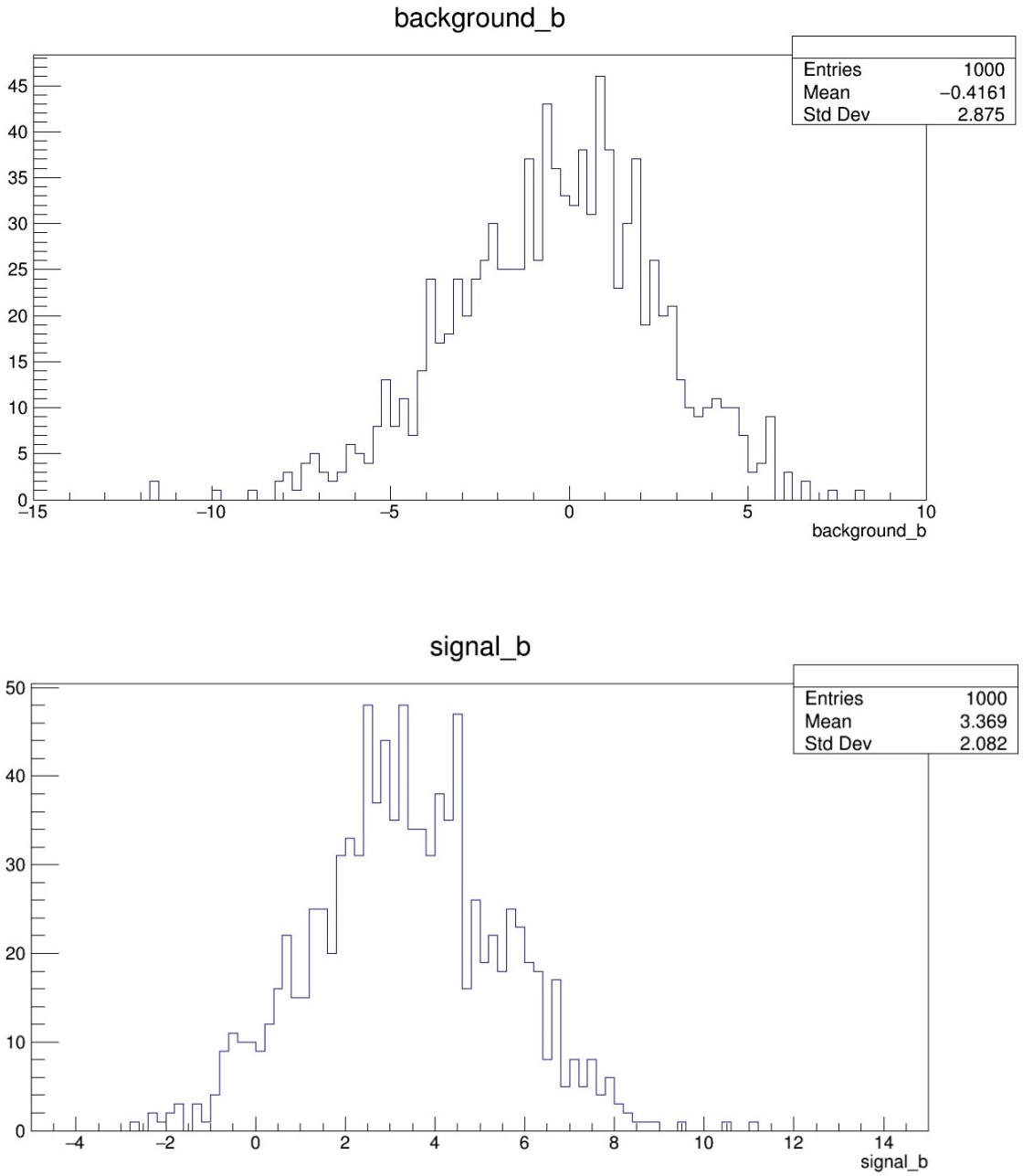

Figure 3: Top: Histogram for background b. Bottom: Histogram for signal b.

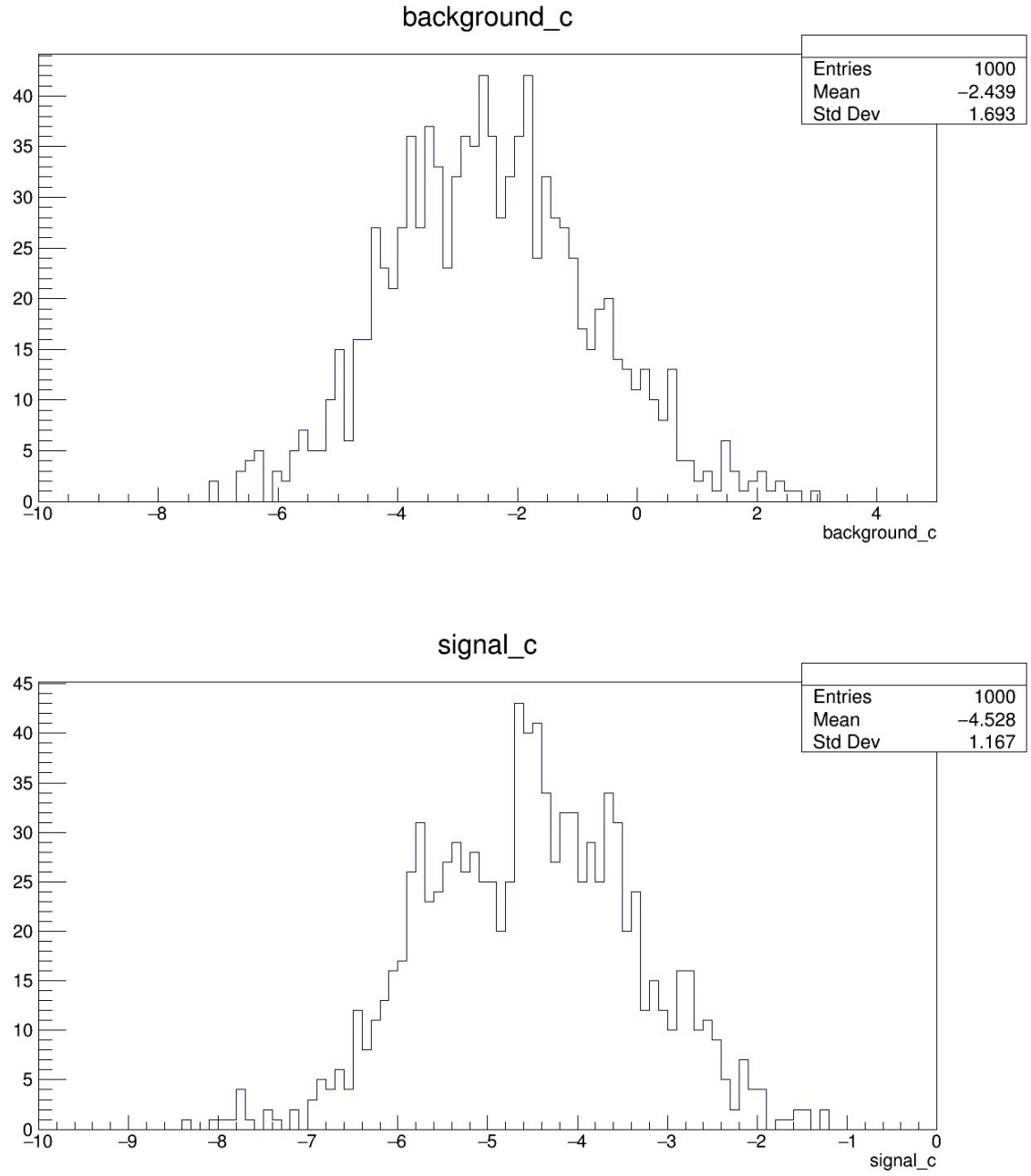

Figure 4: Top: Histogram for background c. Bottom: Histogram for signal c.

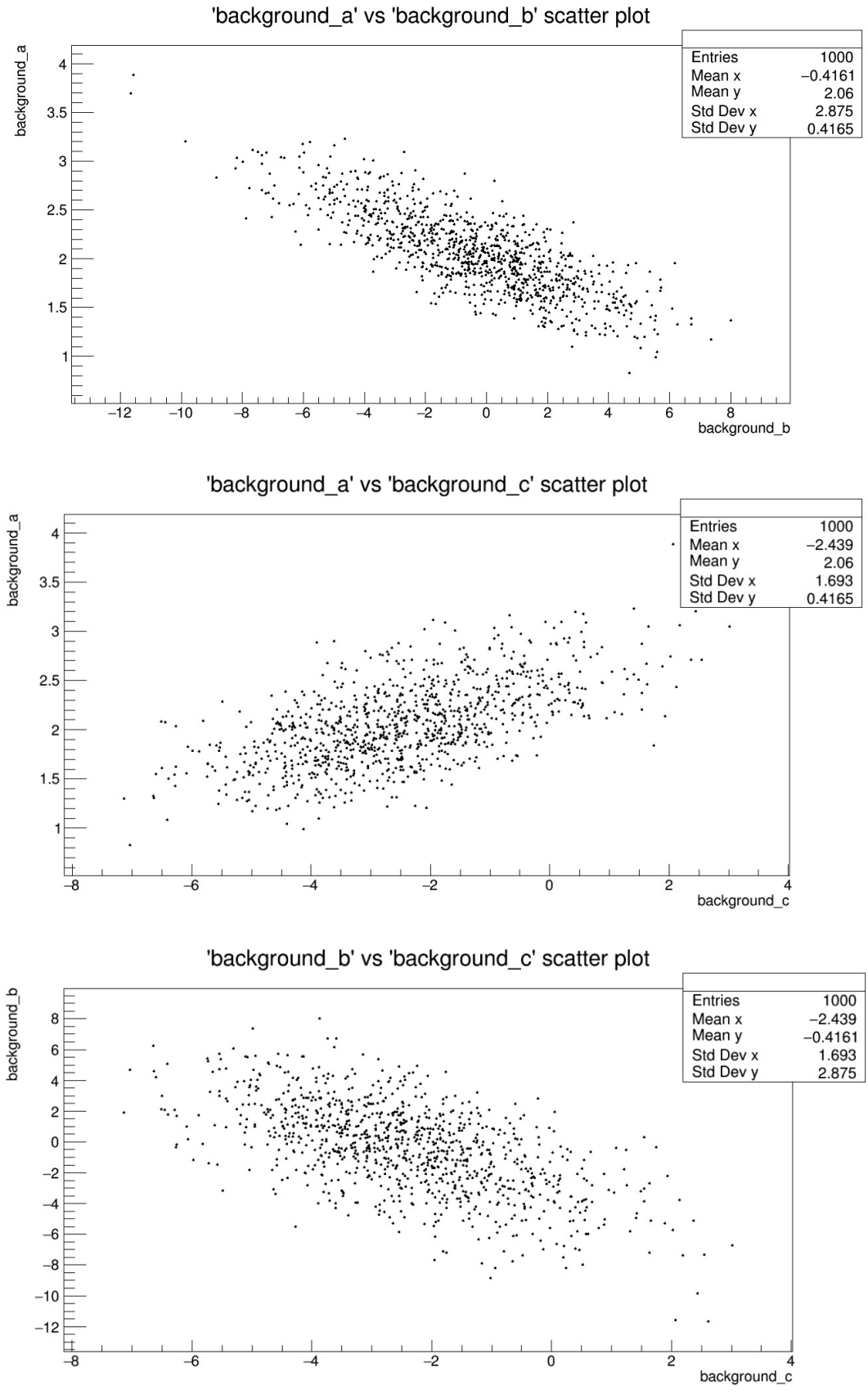

Figure 5: Top: a vs b background scatter plot, Middle: a vs c background scatter plot, Bottom: b vs c background scatter plot.

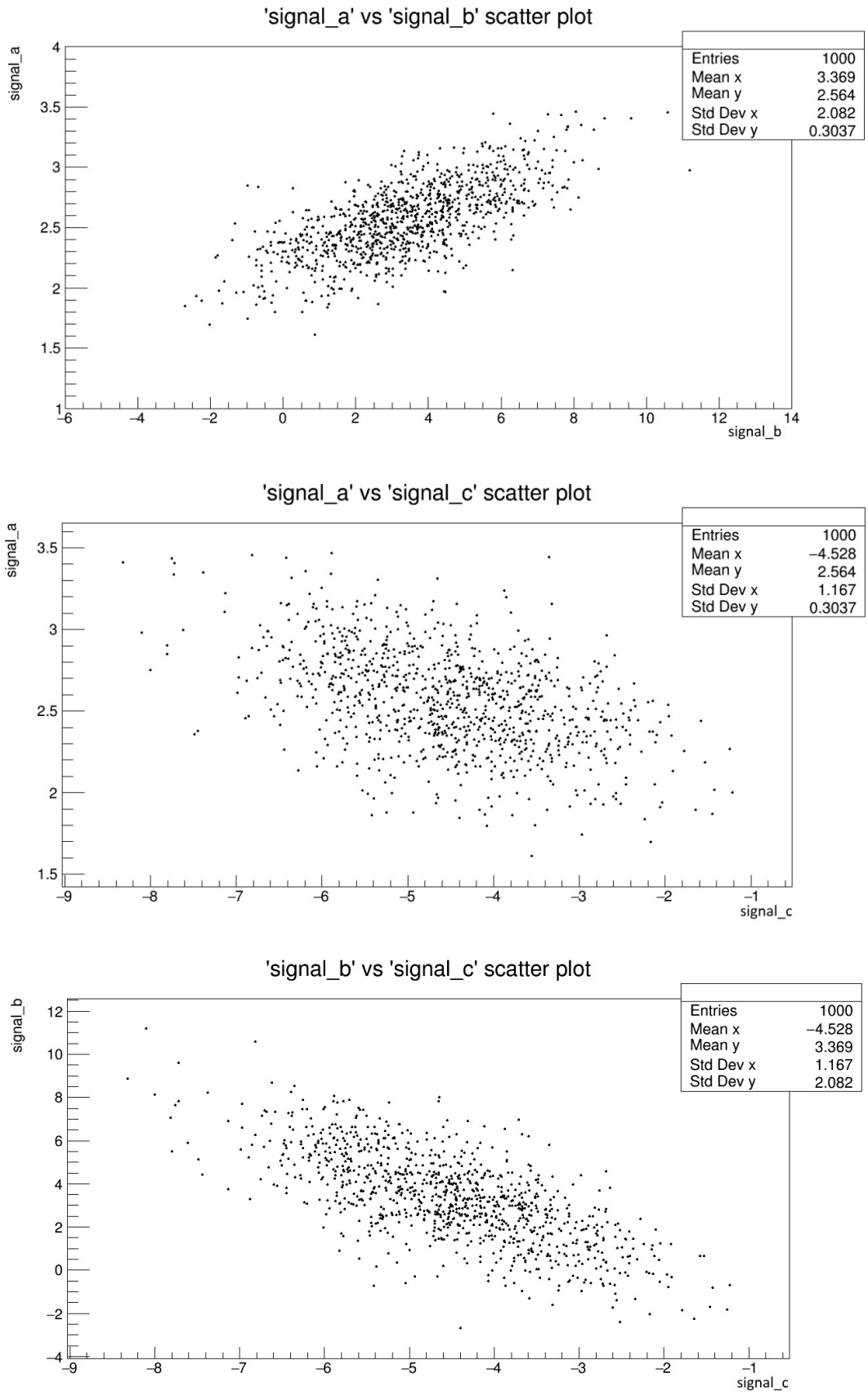

Figure 6: Top: a vs b signal scatter plot, Middle: a vs c signal scatter plot, Bottom: b vs c signal scatter plot.

signal covariance matrix	a	b	c
a	0.092	0.44	-0.16
b	0.44	4.34	-1.71
c	-0.16	-1.71	1.36

Table 1: Signal covariance matrix.

Background covariance matrix	a	b	c
a	0.17	-0.96	0.43
b	-0.96	8.27	-2.93
c	0.43	-2.93	2.87

Table 2: Background covariance matrix.

Signal Correlation matrix	a	b	c
a	1	0.70	-0.47
b	0.70	1	-0.70
c	-0.70	-0.60	1

Table 3: Signal correlation matrix.

Background Correlation matrix	a	b	c
a	1	-0.79	0.60
b	-0.79	1	-0.60
c	0.60	-0.60	1

Table 4: Background correlation matrix.

means	a	b	c
Signal	2.56	3.37	-4.53
Background	2.06	-0.42	-2.44

Table 5: Means for a, b, and c signal, and background.

3 The Fisher Discriminant

The construction of the Fisher discriminant theory is presented below, following the theory in [2]. (From now on I will use index 0 interchangeably with background, and index 1 with signal, so for example: μ_0 is a background mean).

As we saw in equation 1, the Fisher test statistic is a linear combination of the three variables a, b, and c. The objective is to maximize the separation between variable means (or maximize separation between histogram peaks), and we want the peaks as sharply defined as possible, to decrease the amount of overlap between histograms, so we need to decrease variance or increase the reciprocal of variance, this leads to the following definition of the objective function:

$$J(\overrightarrow{k}) = \frac{(\tau_0 - \tau_1)^2}{\Sigma_0^2 + \Sigma_1^2} \tag{2}$$

This is the function to be maximized, where:

$$\tau_i = \int t \; g(t|H_i)dt = \overrightarrow{k}^T \; \overrightarrow{\mu}_i \tag{3}$$

$$\Sigma_i^2 = \int (t - \tau_i)^2 \; g(t|H_i)dt = \overrightarrow{k}^T \; \hat{V}_i \; \overrightarrow{k} \tag{4}$$

i runs over (0,1) the background and signal, $\overrightarrow{\mu}$ is a vector of means for a, b, and c, and $\hat{V}$ is the covariance matrix.

Plugging these into J and setting the derivative with respect to $\overrightarrow{k}$ equal to zero, we find:

$$\overrightarrow{k} \; \propto \; \hat{W}^{-1} \; (\overrightarrow{\mu_0} - \overrightarrow{\mu_1}) \tag{5}$$

where $\hat{W}$ is the sum of the covariance matrices for background and signal

$$\hat{W} = \hat{V_0} + \hat{V_1} \tag{6}$$

The calculations were done using ROOT framework, and NumPy library in python (code in Appendix A), and the relevant quantities were found:

The means and standard deviations of the pds of t for background and signal $\hat{\tau}$, and $\hat{\sigma}$:

$$\hat{\tau} = \begin{bmatrix} \tau_0 \\ \tau_1 \end{bmatrix} = \begin{bmatrix} -6.53 \\ -9.76 \end{bmatrix} \tag{7}$$

$$\hat{\sigma} = \begin{bmatrix} \sigma_0 \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} 0.78 \\ 1.62 \end{bmatrix} \tag{8}$$

$$\hat{W}^{-1} = \begin{bmatrix} 4.12 & 0.12 & -0.12 \\ 0.12 & 0.13 & 0.14 \\ -0.12 & 0.14 & 0.40 \end{bmatrix} \tag{9}$$

$$(\overrightarrow{\mu_0} - \overrightarrow{\mu_1}) = \begin{bmatrix} \mu_{0a} - \mu_{1a} \\ \mu_{0b} - \mu_{1b} \\ \mu_{0c} - \mu_{1c} \end{bmatrix} = \begin{bmatrix} -0.50 \\ -3.79 \\ 2.09 \end{bmatrix} \tag{10}$$

The weights:

$$\overrightarrow{k} = \begin{bmatrix} k_1 \\ k_2 \\ k_3 \end{bmatrix} = \begin{bmatrix} k_a \\ k_b \\ k_c \end{bmatrix} = \begin{bmatrix} -2.80 \\ -0.28 \\ 0.36 \end{bmatrix} \tag{11}$$

4 Signal Extraction

With our weights vector ($\overrightarrow{k}$) we can now define a critical region based on the training data. For each (a, b, c) triplet in the training set we calculate the corresponding $t(\overrightarrow{x})$, and we get the following histogram for t-values of both signal and background:

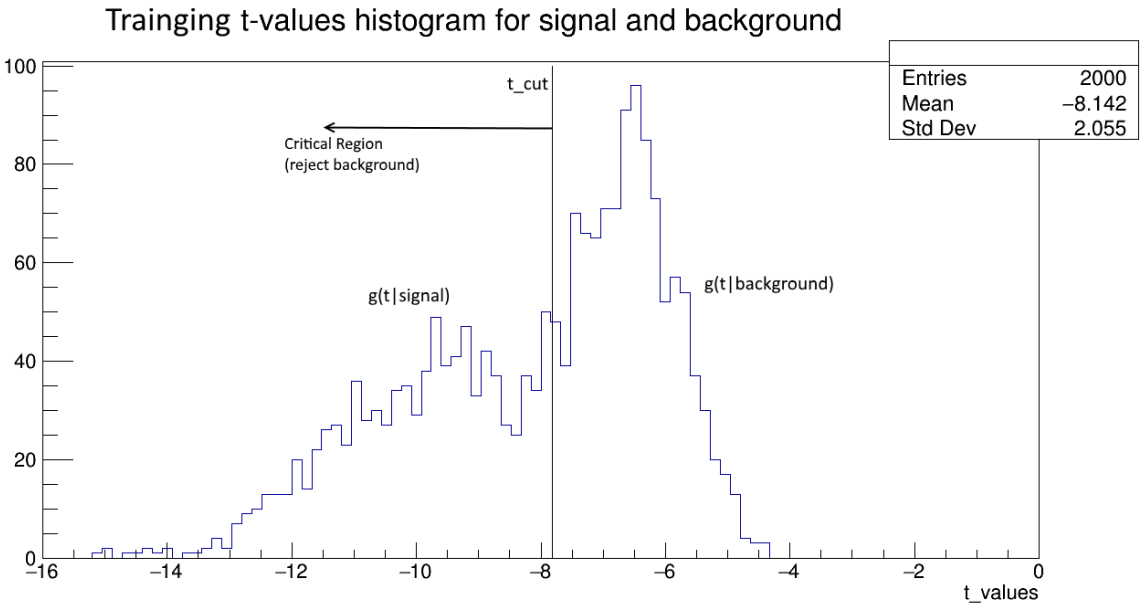

Figure 7: Training data t-values histogram for both background and signal, t_{cut} = -7.813, to the left of which lies the critical region.

We have the background peak to the right, and the signal peak to the left ($\mu_0 > \mu_1$ as we saw in the previous section), with some overlap between the peaks. The cut was found by choosing a significance of $\alpha = 0.05$ then looping over the background t-values and finding the t-value with p-value closest to 0.05 significance, using the background mean and standard deviation ($\hat{\tau}_0$ and $\hat{\sigma}_0$ found earlier):

$$p - value = \Phi(\frac{t - \hat{\tau}_0}{\hat{\sigma}_0}) \tag{12}$$

where Φ is the standard normal ($\mu = 0$, $\sigma = 1$) cumulative distribution function (finds the area under the curve from $-\infty$ to t). t_{cut} was found to be -7.813 .

We can now find the type-1 and type-2 errors (and signal efficiencies ε), by finding the t_{cut} p-value using $\hat{\tau}_1$ and $\hat{\sigma}_1$:

$$type1 \;\; error = \;\; \varepsilon_0 \;\; = \;\; \alpha \;\; = 0.05 \tag{13}$$

$$type2 \;\; error = \;\; \beta \;\; = \;\; 1 - \Phi(\frac{t_{cut} - \hat{\tau}_1}{\hat{\sigma}_1}) = 0.11 \tag{14}$$

$$power \;\; = \;\; \varepsilon_1 \;\; = \;\; 1 - \beta \;\; = 0.89 \tag{15}$$

Where type-1 error indicates probability to reject background even when it is true, and type-2 error indicates probability to reject signal even when it is true, and *power*, is the power of the test to discriminate against the signal. (Note: these efficiencies are related to the ratio of misclassified events to the total number of events, $\varepsilon = \alpha = \frac{N_{miss}}{N_{total}}$ for background, and $1 - \varepsilon = \beta = \frac{N_{miss}}{N_{total}}$ for signal).

Now that we have all relevant quantities we can finally analyze the mystery data. Using the t_{cut} found we can apply that cut to the mystery data, and define a critical region, in which background is rejected, and events are considered to be signal events.

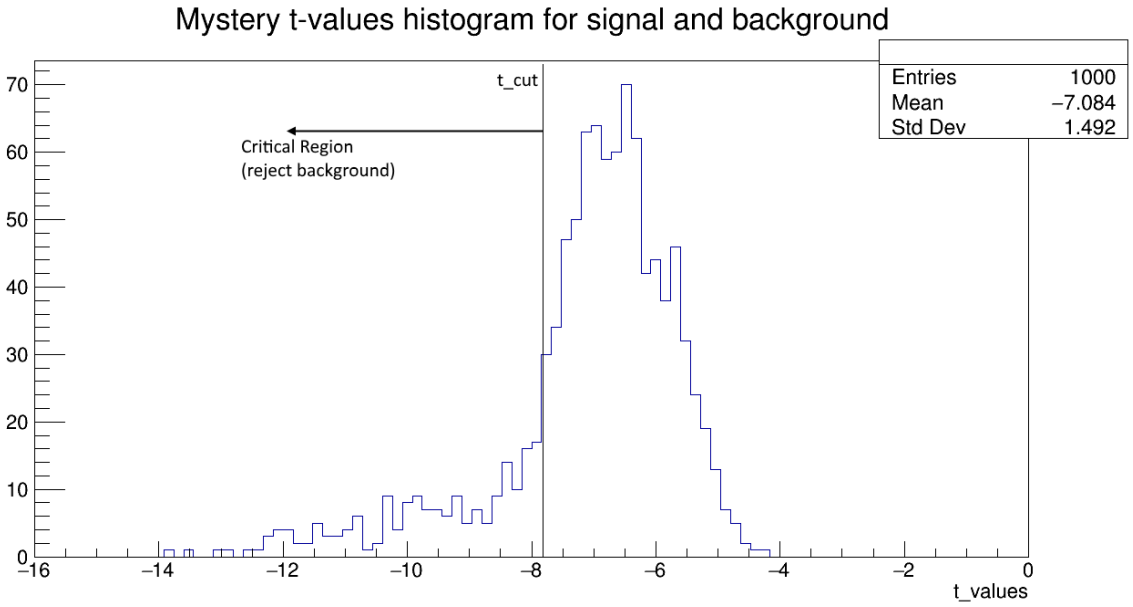

Figure 8: Mystery data t-values histogram, t_{cut} = -7.813, to the left of which lies the critical region.

As we can see in comparison to figure 7, the signal peak in figure 8 is much smaller. Counting the number of events that lie in the critical region ($t < t_{cut}$) we find 189 signal events. Using efficiencies we can say that $\frac{189}{N_{total}} = \varepsilon_1$, so for the signal $N_{total} = 212$, $\therefore N_{miss} = \beta * N_{total} = 24$ of all signal events are misclassified as background.

5 Conclusion/Discussion

In this Project we constructed the Fisher discriminant (which is a linear test statistic) using training data with known (labeled) background, and signal events, and found the critical region for background rejection, we then applied the cut to the mystery data (containing unknown ratios of background to signal), and found the signal by counting the number of events for which background can be rejected (out of 1000 total mystery events). The number of signal events found was N_{signal} = 189 (with an expected 24 other signal events that were misclassified as background).

Some implicit assumptions were made in the construction and analysis using the Fisher discriminant, such as the assumption that the signal points are linearly separable from the background points (by some $2D$ plane in the abc vector space). This is related to the assumption that the a, b, c variables follow a Gaussian pdf. Under this assumption we can say that they are indeed linearly separable, and later in the analysis we can use the standard Normal cumulative function Φ to find the p-values. Another thing to note is that since $t(\overrightarrow{x})$ is constructed as the sum of random variables, as the number of variables increases we can expect the pdf of t, $g(t)$ to approach a Gaussian by the central limit theorem, regardless of the shape of pdf that the variables follow (but only for "large" number of variables, usually taken to be > 30).

If however they didn't follow a Gaussian, then they are not linearly separable and we would need to resort to some other method of classification, such as Support vector machines with non-linear kernels (where the data becomes linearly separable in the kernel space), boosted decision trees, or neural networks.

An example of this can be see below in figure 9 [3], the data on the left is linearly separable, by drawing a line decision boundary the two classes (red and blue) can be separated, whereas for the data on the right, it is not possible to draw a line that can separate both classes, the decision boundary would need to be nonlinear (possibly taking the form of a circle $x_1^2 + x_2^2 = r^2$).

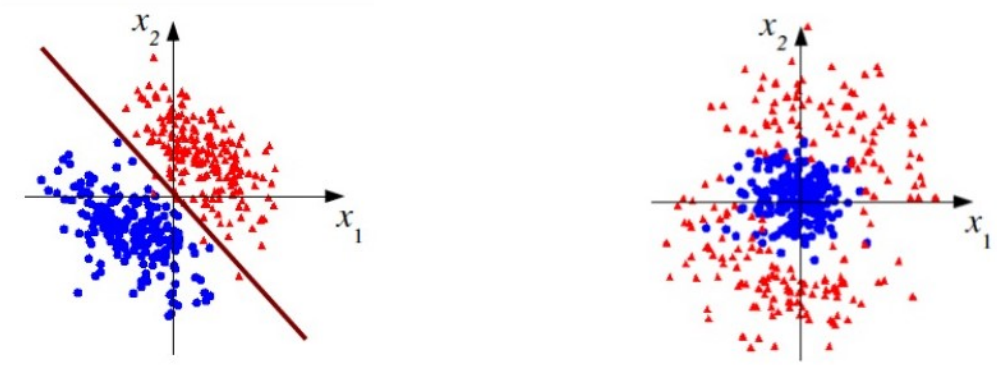

Figure 9: Left: Example of linearly separable data. Right: Example of non-linearly separable data.

Another thing worth mentioning is the choice of significance, which is crucial because it defines the critical region for which background is rejected. There is no standard way to choose this value, but usually it is taken to have a small value (in this case I chose $\alpha = 0.5$), to make it more "difficult" to reject background, because usually the background or null hypothesis is the established hypothesis that we believe in, so we want to make rejecting it in favor of a new hypothesis more difficult (A bit of a tangent: hence our choice of language during hypothesis testing, usually we say either reject the null hypothesis, or we say don't reject the null hypothesis, but we don't say we accept it, implying that it is already accepted, whereas we can accept or reject the alternative hypothesis).

A The Code

The Code used to do the calculations in this project are included on the next four pages.

```
from __future__ import division
import ROOT
import numpy as np
import math

signal_A = []
signal_B = []
signal_C = []

bkg_A = []
bkg_B = []
bkg_C = []

f = ROOT.TFile.Open("Mohamed_training_sets.root")
if f == ROOT.nullptr:
  fatal("Could not open file")

tree_signal = f.Get("signal")
if tree_signal == ROOT.nullptr:
  fatal("Failed accessing signal tree")

tree_bkg = f.Get("background")
if tree_bkg == ROOT.nullptr:
  fatal("Failed accessing background tree")

nsignal_events = tree_signal.GetEntries()
nbkg_events = tree_bkg.GetEntries()  #same as nsignal_events

for ievent in range(nsignal_events):
  tree_signal.GetEntry(ievent)
  tree_bkg.GetEntry(ievent)

  #storing a,b,c in lists

  signal_A.append(tree_signal.a)
  signal_B.append(tree_signal.b)
  signal_C.append(tree_signal.c)

  bkg_A.append(tree_bkg.a)
  bkg_B.append(tree_bkg.b)
  bkg_C.append(tree_bkg.c)


###code used to draw the 1d and 2d histograms, same code was used replacing the variables
for each particular plot
#can1 = ROOT.TCanvas()
##hist1 = ROOT.TH1D("", "background_c", 100, -10, 5)
#hist1 = ROOT.TH2D("", "'background_b' vs 'background_c' scatter plot", 1000, -20, 20, 1000,
20, 20)
##binWidth = (hist1.GetXaxis().GetXmax() - hist1.GetXaxis().GetXmin())/hist1.GetNbinsX()
#
#for i in range(1000):
#  hist1.Fill(bkg_C[i], bkg_B[i])
#hist1.GetXaxis().SetTitle('background_c')
#hist1.GetYaxis().SetTitle('background_b')
#hist1.SetMarkerStyle(20)
#hist1.SetMarkerSize(0.5)
#
#
##for num in bkg_C:
##  hist1.Fill(num)
##hist1.GetXaxis().SetTitle('background_c')
#
##scale = 1/(hist1.Integral()*binWidth)      #Integral() counts number of entries
##hist1.Scale(scale)                         #Normalized with respect to area
#hist1.Draw()



###changing lists to numpy arrays, to calculate covariance matrices:
```

```
signal_A = np.asarray(signal_A)
signal_B = np.asarray(signal_B)
signal_C = np.asarray(signal_C)

bkg_A = np.asarray(bkg_A)
bkg_B = np.asarray(bkg_B)
bkg_C = np.asarray(bkg_C)

signal_means = np.array([np.mean(signal_A), np.mean(signal_B), np.mean(signal_C)])
bkg_means = np.array([np.mean(bkg_A), np.mean(bkg_B), np.mean(bkg_C)])


sigVaa = np.cov(signal_A, signal_B)[0][0]
sigVbb = np.cov(signal_A, signal_B)[1][1]
sigVab = np.cov(signal_A, signal_B)[0][1]

sigVcc = np.cov(signal_A, signal_C)[1][1]
sigVac = np.cov(signal_A, signal_C)[0][1]

sigVbc = np.cov(signal_B, signal_C)[0][1]

#now we can define the covariance matrix for the signal
V_signal = np.array([[sigVaa, sigVab, sigVac], [sigVab, sigVbb, sigVbc], [sigVac, sigVbc,
sigVcc]])


# now repeating but for background:
bkgVaa = np.cov(bkg_A, bkg_B)[0][0]
bkgVbb = np.cov(bkg_A, bkg_B)[1][1]
bkgVab = np.cov(bkg_A, bkg_B)[0][1]

bkgVcc = np.cov(bkg_A, bkg_C)[1][1]
bkgVac = np.cov(bkg_A, bkg_C)[0][1]

bkgVbc = np.cov(bkg_B, bkg_C)[0][1]

V_bkg = np.array([[bkgVaa, bkgVab, bkgVac], [bkgVab, bkgVbb, bkgVbc], [bkgVac, bkgVbc,
bkgVcc]])

###the W matrix that will be used to define the fisher discriminant weights:
W = V_bkg + V_signal

Winv = np.linalg.inv(W)

###we can now find the test statistic t weights (so t = K_transpose . x):

K = np.dot(Winv, (bkg_means - signal_means))




###now we find the mean and std dev of t:

tsignal_mean = np.dot(K, signal_means)
tsignal_var = np.dot(K, np.dot(V_signal, K))
tsignal_stdev = tsignal_var**0.5


tbkg_mean = np.dot(K, bkg_means)
tbkg_var = np.dot(K, np.dot(V_bkg, K))
tbkg_stdev = tbkg_var**0.5

t_signal =[]
t_bkg = []

for i in range(1000):
  t_signal.append(K[0]*signal_A[i] + K[1]*signal_B[i] + K[2]*signal_C[i])
  t_bkg.append(K[0]*bkg_A[i] + K[1]*bkg_B[i] + K[2]*bkg_C[i])

t_all = t_signal + t_bkg
```

```
###we can now define our significance level, and use that to find tcut, so apply to mystery
data:
###note mean bkg (our null hypothesis H0 backgorund mean is larger than mean signal our H1):

alpha = 0.05
tcut = []
for j in t_bkg:
  if abs(ROOT.Math.normal_cdf(((j-tbkg_mean)/tbkg_stdev)) - alpha) < 0.0005:
    tcut.append(j)

#this gives only one element, the closest to 0.05 cutoff
tcut = tcut[0]




###calculating efficiencies:

bkg_eff = ROOT.Math.normal_cdf((tcut - tbkg_mean)/tbkg_stdev)  #very close to 0.05

beta = 1 - ROOT.Math.normal_cdf((tcut - tsignal_mean)/tsignal_stdev)

signal_eff = 1 - beta

###missclassification ratios, these are about the same as the efficiencies:
bkg_miss = []
signal_miss = []

for i in t_bkg:
  if i < tcut:
    bkg_miss.append(i)

for i in t_signal:
  if i > tcut:
    signal_miss.append(i)

bkg_ratio = len(bkg_miss)/len(t_bkg)

signal_ratio = len(signal_miss)/len(t_signal)




###I now have a value for tcut, below which any events are considered signal, we can apply
this to the mystery data

mystery_A = []
mystery_B = []
mystery_C = []

m = ROOT.TFile.Open("Mohamed_mystery_data.root")
if m == ROOT.nullptr:
  fatal("Could not open file")

tree_mystery = m.Get("data")
if tree_mystery == ROOT.nullptr:
  fatal("Failed accessing mystery tree")

n_mystery = tree_mystery.GetEntries()

for i in range(n_mystery):
  tree_mystery.GetEntry(i)

  mystery_A.append(tree_mystery.a)
  mystery_B.append(tree_mystery.b)
  mystery_C.append(tree_mystery.c)
```

```
t_mystery = []
signal_mystery = []

for i in range(1000):
  t_mystery.append(K[0]*mystery_A[i] + K[1]*mystery_B[i] + K[2]*mystery_C[i])



#can1 = ROOT.TCanvas()
#hist1 = ROOT.TH1D("", "Mystery t-values histogram for signal and background", 100, -16, 0)
##binWidth = (hist1.GetXaxis().GetXmax() - hist1.GetXaxis().GetXmin())/hist1.GetNbinsX()
#
#for num in t_mystery:
#  hist1.Fill(num)
#
#hist1.GetXaxis().SetTitle("t_values")
##scale = 1/(hist1.Integral()*binWidth)      #Integral() counts number of entries
##hist1.Scale(scale)                         #Normalized with respect to area
#hist1.Draw()
#
#line = ROOT.TLine(-7.8126, 0, -7.8126, 73)
#line.Draw('SAME')


for i in t_mystery:
  if i < tcut:
    signal_mystery.append(i)

print "number of signal events is " , len(signal_mystery)," with around
",round(beta*(len(signal_mystery)/signal_eff)), " signal events missclassified as background"
```

References

[1] "Dynamic stardust",

What is dimensionality reduction?,

https://datascience.stackexchange.com/questions/130/what-is-dimensionality-reduction-what-is-the-difference-between-feature-selecti

[2] G. Cowan,

Statistical Data Analysis

[3] D. Gillberg,

PHYS 5002 lecture 4 slides